

On Solving the Maximum k -club Problem

Andreas Wotzlaw

Institut für Informatik, Universität zu Köln, Weyertal 121, D-50931 Köln, Germany
wotzlaw@informatik.uni-koeln.de

Abstract. Given a simple undirected graph G , the maximum k -club problem is to find a maximum-cardinality subset of nodes inducing a subgraph of diameter at most k in G . This NP-hard generalization of clique, originally introduced to model low diameter clusters in social networks, is of interest in network-based data mining and clustering applications. We give two MAX-SAT formulations of the problem and show that two exact methods resulting from our encodings outperform significantly the state-of-the-art exact methods when evaluated both on sparse and dense random graphs as well as on diverse real-life graphs from the literature.

Keywords: maximum k -club problem, clique relaxation, cohesive subgroups, partial max-sat problem, satisfiability, exact algorithm

1 Introduction

Let $G = (N, E)$ be a simple undirected graph with the set N of n nodes and the set E of m edges. The length of a shortest path between two nodes i and j in G is denoted by $\text{dist}_G(i, j)$, whereas $d_G := \max_{i, j \in N} \text{dist}_G(i, j)$ is the *diameter* of G . For a nonempty subset of nodes $S \subseteq N$, $G[S]$ denotes the subgraph $(S, E(S))$ of G induced by S on G , where $E(S)$ are edges of E with both end nodes in S . If every pair of nodes $i, j \in S$ is connected in $G[S]$ by at least one path with at most k edges, in other words, $d_{G[S]}$ is at most k , then S is called a k -club of G .

The *maximum k -club problem*, MkCP, consists in finding a maximum cardinality k -club in G . We denote the cardinality of a maximum k -club in G by $\omega_k(G)$, referred to also as the *k -club number* of G . A k -club is regarded as diameter-based relaxation of *clique* [24]. Recall, a clique C in G is a subset of N such that the subgraph $G[C]$ of G is complete, i.e., $d_{G[C]} = 1$. Hence, for $k = 1$, the definition of k -club is equivalent to that of clique.

The notion of k -club was introduced in social network analysis [18] as an alternative way to model tightly linked groups of actors (e.g., people, companies, web communities or sites), referred to as *cohesive subgroups* [21]. In those groups every member is related to all other members either directly or via other members. Although cliques are useful for modeling high-density communities [7], they appear to be too restrictive to represent real-life groups where rarely all members are connected directly. Here, the idea of k -club can be used instead to model low-diameter clusters in graphs. It finds its application in graph-based data mining in social, biological, financial, and communication networks [1, 5, 23].

Related work. MkCP is computationally challenging. Bourjolly et al. [8] established the NP-hardness of MkCP, even for fixed $k > 1$, and proposed an exact branch-and-bound algorithm for it. Balasundaram et al. [5] showed that MkCP remains NP-hard even when restricted to graphs of fixed diameter. Unlike cliques, the k -club model is of nonhereditary nature [18], meaning that every subset of a k -club is not necessarily a k -club itself. An important manifestation of this property is the intractability of testing maximality of k -clubs, demonstrated by Mahdavi Pajouh and Balasundaram [17]. They also developed a branch-and-bound technique B&B for MkCP using the k -coloring number as an upper bound. For fixed $k \geq 2$, Asahiro et al. [2] proved that MkCP is inapproximable within a factor of $n^{\frac{1}{2}-\epsilon}$ for any $\epsilon > 0$, unless $P=NP$. MkCP is fixed-parameter tractable when parameterized by solution size as shown by Schäfer et al. [20]. In [14,15], Hartung et al. gave recently a systematic classification of the complexity of M2CP with respect to several structural graph parameters like, e.g., feedback edge set size, as well as a new well-performing parameterized algorithm for M2CP. Moreover, Schäfer [19] demonstrated that M2CP on bipartite graphs can be solved in $O(n^5)$ time, whereas MkCP on trees and interval graphs needs $O(kn^2)$ and $O(n^2)$ time, respectively. Chang et al. [11] proved recently that MkCP can be solved exactly in $O^*(1.62^n)$ time, where O^* hides factors polynomial in n . The first polyhedral results for 2-club polytope were given in [5]. While MkCP has a compact Boolean integer programming (BIP) formulation for $k = 2$, the formulations proposed for $k \geq 3$ in [5,9] need exponentially many variables. Alternative BIP formulations for $k = 3$ were explored by Almeida and Carvahlo [1]. The first polynomial-size BIP formulation for a general k using $O(kn^2)$ variables and constraints was given by Veremyev and Boginski [24]. Further, Chang et al. [11] implemented a branch-and-bound algorithm for MkCP using a new heuristic IDROP for finding initial lower bounds. Finally, Shahinpour and Butenko [23] presented a well-performing exact branch-and-bound method for MkCP using variable neighborhood search for lower bounding.

Our contribution. In this paper, we present a new exact approach for MkCP. To this end, we give first in Section 2 two propositional-logic-based formulations of MkCP. In both cases, we encode MkCP on graph G as an instance of the PARTIAL MAX-SAT problem [10] of some propositional formula in conjunctive normal form with a mandatory part of clauses that must be satisfied for the solution to be reasonable, and a second part of clauses of length 1 (1-clauses), such that a truth assignment must satisfy as many of them as possible. In an optimal solution of such a PARTIAL MAX-SAT instance, the number of satisfied 1-clauses is then equal to $\omega_k(G)$, and the Boolean variables assigned by the truth assignment to 1 indicate the nodes of G included in an optimal k -club of G . Our first satisfiability-based formulation of MkCP needs $O(n^{k-1})$ variables and clauses, whereas for the second one $O(kn^2)$ variables and $O(kn^3)$ clauses suffice.

According to the experimental evaluation for $k \in \{2, 3, 4\}$ (i.e., for typical values of k from the literature [1,5,23]) given in Section 3, our exact methods SatMC1 and SatMC2 for MkCP incorporating the encodings from Section 2, when compared with a straightforward exact BIP-based approach using the problem

formulation described in [1,24], as well as with two well-performing specialized exact branch-and-bound methods VNS [23] and B&B [17], demonstrate clearly their practical strength by outperforming the other three methods considerably. Also, they offer a simple yet effective alternative for finding good-quality approximate solutions for MkCP, as the numerical results for our both methods show. Finally, in Section 4 we conclude our work and state some open questions.

2 Satisfiability-based Formulation of MkCP

Preliminaries. Let CNF denote the set of propositional formulas in conjunctive normal form over a set V of Boolean variables. Each variable $x \in V$ induces a positive literal (variable x), or a negative literal (negated variable \bar{x}). Each formula $C \in \text{CNF}$ is regarded as a *set* of its clauses. Similarly, a clause is considered as a *set* of its literals. A clause is termed a *k-clause*, for some integer $k > 0$, if it contains exactly k literals. We denote by $V(C)$ the set of variables occurring in formula C . The satisfiability problem (SAT) asks whether formula C is *satisfiable*, i.e., whether there is a truth assignment $t : V(C) \rightarrow \{0, 1\}$ setting at least one literal in each clause of C to 1, whereas for every $x \in V$ it holds $t(x) = 1 - t(\bar{x})$. Given a formula $C \in \text{CNF}$, the optimization version MAX-SAT searches for a truth assignment t satisfying as many clauses of C as possible, whereas in its PARTIAL variant some clauses (called *hard*) must be satisfied.

Our Method. We only consider simple undirected graphs $G = (N, E)$ with $N = \{1, \dots, n\}$ and $m := |E|$. Each node is referred to by its number. Let $A := (a_{ij})$ be the adjacency matrix of G , where the values a_{ij} 's are regarded as constant truth values 0 and 1, such that $a_{ij} = 1$ iff an edge $\{i, j\} \in E$, for $1 \leq i, j \leq n$.

We are now ready to give our two PARTIAL MAX-SAT formulations of MkCP on graph G for an integer $k > 1$. For this purpose, we define for every node $i \in N$ a Boolean variable x_i , such that $x_i = 1$ if and only if i belongs to a specific *k-club* of G . We proceed next in two steps. We define first a CNF formula C_S ensuring the optimality, i.e., the maximum cardinality, of a solution $S \subseteq N$ to MkCP on G . In the second step, we show the construction of two CNF formulas, C_H and D_H , for the first and the second formulation, respectively, both consisting only of hard clauses and ensuring the correctness of a solution S to MkCP, i.e., S is a *k-club* in G . The unions $C_S \cup C_H$ and $C_S \cup D_H$ will give finally the first and the second PARTIAL MAX-SAT encoding of MkCP on G , respectively.

The formula C_S consists of 1-clauses solely and is defined as follows:

$$C_S := \{\{x_1\}, \dots, \{x_n\}\}.$$

Now we construct C_H for the first encoding as follows:

$$C_H := \bigcup_{i=1}^{n-1} \bigcup_{j \in \{i+1, \dots, n\} | a_{ij}=0} \{C_{ij}\}, \quad \text{where} \quad C_{ij} := \{\bar{x}_i, \bar{x}_j\} \cup \bigcup_{l=1}^{k-1} C_{ij}^l$$

and

$$C_{ij}^l := \bigcup_{r_1 \in N_*} \bigcup_{r_2 \in N_1} \dots \bigcup_{r_l \in N_{l-1}} \{x_{r_1} \wedge x_{r_2} \wedge \dots \wedge x_{r_l} \mid a_{ir_1} \wedge a_{r_1 r_2} \wedge \dots \wedge a_{r_l j} = 1\},$$

where $N_* := N \setminus \{i, j\}$ and $N_p := N_* \setminus \{r_1, \dots, r_p\}$, for $p = 1, \dots, l-1$. Note, that each conjunction $x_{r_1} \wedge x_{r_2} \wedge \dots \wedge x_{r_l}$ in C_{ij}^l together with nodes i, j corresponds to a path of length $l+1$ from i to j in G . Due to N_* and N_p in the definition of C_{ij}^l , no paths with cycles can be generated, what may result in a tighter encoding. However, for the correctness of C_H , these restrictions of N are not necessary.

To finish our construction, C_{ij}^l , for $l \geq 2$, has to be transformed into a clause. For this, we replace each occurrence of $x_{r_1} \wedge x_{r_2} \wedge \dots \wedge x_{r_l}$ in C_{ij}^l , for all $i, j \in N$, with a new Boolean variable $y_{r_1 \dots r_l}$ and define $l+1$ additional clauses

$$\{\bar{y}_{r_1 \dots r_l}, x_{r_1}\}, \{\bar{y}_{r_1 \dots r_l}, x_{r_2}\}, \dots, \{\bar{y}_{r_1 \dots r_l}, x_{r_l}\}, \{\bar{x}_{r_1}, \bar{x}_{r_2}, \dots, \bar{x}_{r_l}, y_{r_1 \dots r_l}\},$$

expressing after some elementary transformations the logical equivalence

$$x_{r_1} \wedge x_{r_2} \wedge \dots \wedge x_{r_l} \leftrightarrow y_{r_1 \dots r_l}.$$

Clearly, for $k = 2$, we need $O(n)$ variables and $O(n^2)$ clauses. However, for $k > 2$, C_H requires, in consequence of the transformation of C_{ij}^l into a clause, $O(n^{k-1})$ variables and clauses. Note, that C_{ij} is generated only if $a_{ij} = 0$.

Let t be a truth assignment satisfying C_H , and S the nodes selected by t , i.e., $S = \{i \in N \mid t(x_i) = 1\}$. For the correctness of C_H , it suffices to show which conditions do hold in $G[S]$, if a pair of distinct nodes $i, j \in N$ belongs to S , implying $t(x_i) = t(x_j) = 1$. Obviously, only the case $a_{ij} = 0$ need to be considered. In that case, C_{ij} can be satisfied if and only if there exists at least one conjunction $x_{r_1} \wedge \dots \wedge x_{r_l}$ in C_{ij}^l (or, equivalently, at least one variable $y_{r_1 \dots r_l}$ together with the corresponding clauses after the transformation of C_{ij}^l in a clause given above), for some $l \in \{1, \dots, k-1\}$, satisfied by t , i.e., $t(x_{r_1}) = \dots = t(x_{r_l}) = 1$, implying that nodes r_1, \dots, r_l belong to S , too. Consequently, the nodes i and j are connected in $G[S]$ via l many nodes r_1, \dots, r_l on a path from i to j in $G[S]$. Thus, $d_{G[S]} \leq k$ holds for S specified by t and, by the definition of k -club, we conclude that S is a k -club in G if and only if t satisfies C_H .

Finally, observe that solving MkCP on G is equivalent in terms of propositional calculus to determining a truth assignment satisfying C_H and maximizing the number τ of satisfied clauses in C_S . Clearly, τ corresponds to $\omega_k(G)$, thus completing the description of our first satisfiability-based formulation of MkCP. Note that this formulation works trivially for $k = 1$.

Theorem 1. *Let G be a simple undirected graph, k some positive integer, and $t : V(C_S \cup C_H) \rightarrow \{0, 1\}$ a truth assignment satisfying C_H and maximizing the number τ of satisfied clauses in C_S . Then τ is equal to $\omega_k(G)$. Moreover, for $k > 2$, $C_S \cup C_H$ contains $O(n^{k-1})$ Boolean variables and clauses.*

The formulation above is in the worst case of an exponential size and requires explicit enumeration of all paths of length at most k between all pairs of nodes.

Nevertheless, for typical values of k [1,5,23], C_H is of reasonable size. For $k = 2$, we need only n variables and $n(n+1)/2 - m$ clauses (mostly 2-clauses for sparse graphs). For $k = 3$, at most $n + m$ variables and $n(n+1)/2 + 2m$ clauses suffice.

For $k > 3$, our second formulation of MkCP, $C_S \cup D_H$, is substantially smaller than the first one, as we shall show it now by constructing the CNF formula D_H . For this, we introduce for every pair of distinct nodes $i, j \in N$ and $l = 2, \dots, k$ a new Boolean variable v_{ij}^l , such that $v_{ij}^l = 1$ if and only if there exists at least one path of length at most l from node i to node j in the subgraph $G[S]$ induced by the nodes of a k -club S of G . Initially, for $l = 2$, we can write

$$v_{ij}^2 \leftrightarrow x_i \wedge x_j \wedge \left(\bigvee_{r=1}^n a_{ir} \wedge a_{rj} \wedge x_r \right),$$

what after some elementary transformations is equivalent to the CNF formula $D_{ij}^2 := \{ \{ \bar{v}_{ij}^2, x_i \}, \{ \bar{v}_{ij}^2, x_j \}, \{ \bar{v}_{ij}^2, x_{r_1}, \dots, x_{r_p} \}, \{ \bar{x}_i, \bar{x}_j, v_{ij}^2, \bar{x}_{r_1} \}, \dots, \{ \bar{x}_i, \bar{x}_j, v_{ij}^2, \bar{x}_{r_p} \} \}$ where the nodes $r_1, \dots, r_p \in \{ r \in N \mid a_{ir} \wedge a_{rj} = 1 \}$. If no such a node exists, then we set $D_{ij}^2 := \{ \{ \bar{v}_{ij}^2 \} \}$.

For $l \geq 3$, v_{ij}^l can be defined recursively as

$$v_{ij}^l \leftrightarrow x_i \wedge \left(\bigvee_{r=1}^n a_{ir} \wedge v_{rj}^{l-1} \right),$$

what again after some transformations is equivalent to the CNF formula

$$D_{ij}^l := \left\{ \{ \bar{v}_{ij}^l, x_i \}, \{ \bar{v}_{ij}^l, v_{r_1j}^{l-1}, \dots, v_{r_pj}^{l-1} \}, \{ \bar{x}_i, v_{ij}^l, \bar{v}_{r_1j}^{l-1} \}, \dots, \{ \bar{x}_i, v_{ij}^l, \bar{v}_{r_pj}^{l-1} \} \right\},$$

where $r_1, \dots, r_p \in \{ r \in N \mid a_{ir} = 1 \}$. If no such a node exists, then $D_{ij}^l := \{ \{ \bar{v}_{ij}^l \} \}$.

Finally, we define

$$D_H := \bigcup_{i \in N} \bigcup_{j \in N \setminus \{i\} \mid a_{ij}=0} D_{ij}, \quad \text{where} \quad D_{ij} := \{ \{ \bar{x}_i, \bar{x}_j, v_{ij}^2, \dots, v_{ij}^k \} \} \cup \bigcup_{l=2}^k D_{ij}^l.$$

Observe first that D_{ij} has to be generated only if $a_{ij} = 0$. Moreover, to encode D_H for $k > 1$, we need $O((k-1)n^2)$ variables and $O((k-1)n^3)$ clauses. Thus, the encoding size remains polynomial in the input size. Now, similarly to C_{ij} , for every pair of distinct nodes $i, j \in N$, the existence of a satisfying truth assignment t for D_{ij} with $t(x_i) = t(x_j) = 1$ implies that $\text{dist}_{G[S]}(i, j) \leq k$ for $S \subseteq N$ specified by t . Hence, S is a k -club in G if and only if t satisfies D_H .

Finally, note that solving MkCP on G is equivalent to determining a truth assignment satisfying D_H and maximizing the number of satisfied clauses in C_S , completing the description of our second PARTIAL MAX-SAT formulation of MkCP. Obviously, this formulation works fine also for $k = 1$.

Theorem 2. *Let G be a simple undirected graph, k some positive integer, and $t : V(C_S \cup D_H) \rightarrow \{0, 1\}$ a truth assignment satisfying D_H and maximizing the number τ of satisfied clauses in C_S . Then τ is equal to $\omega_k(G)$. Moreover, for $k > 1$, $C_S \cup D_H$ contains $O(kn^2)$ Boolean variables and $O(kn^3)$ clauses.*

3 Comparative Evaluation

Experimental Setup. The goal of our experiments was to evaluate, for typical values of $k \in \{2, 3, 4\}$ from the literature [1,5,23], the performance of two exact methods for $MkCP$, **SatMC1** and **SatMC2**, implemented in C++ according to the first and the second encoding from Section 2, respectively. We tested our methods against a BIP-based approach **IPMC** using $MkCP$ formulations from [1,24], and two state-of-the-art exact methods: the hybrid algorithm for $MkCP$ from [23], denoted here by **VNS**, and the branch-and-bound technique **B&B** [17]. To make the study better comparable with the previous results, we use the same C++ implementations of **VNS** and **B&B** as the ones being tested in [23].

For solving the **PARTIAL MAX-SAT** instances produced by **SatMC1** and **SatMC2**, we applied a complete **MAX-SAT** solver clasp 2.1.3 [12], an example of a modern **SAT** solver. It extends the backtrack search procedure **DPLL** [6], commonly used for **SAT**-solving, with efficient conflict-driven clause learning (**CDCL**), lazy data structures, deletion policies for learned clauses, and periodical restarts of the search procedure, among others. For more details on the key techniques of **DPLL**- and **CDCL**-based **SAT**-solving, we refer to [6]. In **IPMC**, for solving **BIP**-instances we use **CPLEX** 12.1 [16]. All tests were run on a machine with Intel Xeon E5410 2.33 GHz processor running a 64-bit Linux 3.2.51 with 32GB RAM. All programs (solvers) were run with default call parameters in a single-threaded mode with only one CPU core permitted.

There were two sets of graph instances being tested. The first set contained 12 connected simple graphs from the 10th **DIMACS Implementation Challenge** [3]. We used them to test the methods on some real-life networks ranging from small and dense ones to large and sparse ones (see Table 1). The graphs of the second set were generated randomly by the algorithm proposed by Gendreau et al. [13]. This generalization of the classical uniform random graph generator has earlier

Table 1. Statistics on **DIMACS** instances. Here, n and m give the number of nodes and edges, d the edge density, and $\omega_2, \omega_3, \omega_4$ the club numbers computed by **SatMC**{1, 2}.

Instance	n	m	d	ω_2	ω_3	ω_4
adjnoun	112	425	0.0684	50	82	107
football	115	613	0.0935	16	58	115
jazz	198	2742	0.1406	103	174	192
celegansm	453	2025	0.0198	238	371	432
email	1133	5451	0.0085	72	212	651
polblogs	1490	16715	0.0151	352	776	1127
add20	2395	7462	0.0022	124	671	1454
data	2851	15093	0.0037	18	32	52
3elt	4720	13722	0.0012	10	16	27
add32	4960	9462	0.0008	32	99	268
hep-th	8361	15751	0.0006	51	120	344
whitaker3	9800	28989	0.0006	9	15	23

been used for testing new methods for MkCP [1,8,17,23]. The edge density of the graphs produced by this method was controlled by two parameters a and b ($0 \leq a \leq b \leq 1$). The *expected* edge density D is $(a + b)/2$, and the node degree variance (NDV) increases with the increase in $b - a$. In our tests, we used *connected* graphs with $n = 100, 150$, and 200 and $D = 0.035, 0.05, 0.1, 0.15$, and 0.2 , i.e., from the range of challenging instances according to [1,8]. For each graph size n and density D , we generated 10 samples with minimum NDV ($a = b = D$) and 10 samples with maximum NDV ($a = 0, b = 2D$), denoted in the following by min and max, respectively. As in [1] and in contrast to [17,23], we decided to reject samples with more than one component since for them the number of nodes would be misleading. It would correspond mostly to the cardinality of the largest component. Though, the maximum k -club may or may not be located in the largest or the most dense component as indicated already in [17].

The running time limit for solving each instance tested was set to 3600 seconds for each method. If an instance could not be solved into optimality within that time, the computation has been terminated, the best solution computed so far (i.e., a lower bound for the optimum), as well as the upper bound have been recorded, and an optimality gap, $(\text{upper bound} - \text{best solution size})/(\text{upper bound})$, has been reported. The CNF formulas generated by our methods for the DIMACS instances included up to 3.5 millions variables and 50 millions clauses. **SatMC2** required in most cases, and in particular for $k \in \{2, 3\}$, up to 10 times more variables than **SatMC1** did. For sparse graphs, the number of clauses required by both methods was similar. The time **SatMC** $\{1, 2\}$ needed for the generation of CNF formulas is *included* in the running times given below.

Results for real-life graphs. Tables 2, 3, and 4 show the running times (in seconds) and the optimality gaps for the DIMACS instances solved by **IPMC**, **B&B**, **VNS**, and **SatMC** $\{1, 2\}$ for $k \in \{2, 3, 4\}$, respectively. For **IPMC** no optimality gaps are

Table 2. Computational results of solving MkCP for $k = 2$ on DIMACS instances.

Instance	IPMC		B&B		VNS		SatMC1		SatMC2	
	time (s)		time (s)	gap	time (s)	gap	time (s)	gap	time (s)	gap
adnoun	0.28		0.16	0	0.52	0	0.01	0	0.02	0
football	8.91		0.74	0	0.76	0	0.02	0	0.03	0
jazz	3.98		10.1	0	26.8	0	0.06	0	0.21	0
celegansm	39.7		26.3	0	29.5	0	0.43	0	2.3	0
email	>3600		39.2	0	39.6	0	16.1	0	24.1	0
polblogs	74.9		1351	0	1359	0	46.9	0	90.7	0
add20	63.9		126.6	0	141.6	0	108.2	0	129.9	0
data	>3600		>3600	0.18	>3600	0.22	28.8	0	36.3	0
3elt	>3600		>3600	0.29	>3600	0.29	105.9	0	87.3	0
add32	>3600		59.5	0	136.5	0	80.1	0	50.2	0
hep-th	>3600		199.2	0	242.5	0	624.1	0	273.3	0
whitaker3	>3600		>3600	0.36	>3600	0.36	1091	0	1045	0

Table 3. Computational results of solving $MkCP$ for $k = 3$ on DIMACS instances.

Instance	IPMC	B&B		VNS		SatMC1		SatMC2	
	time (s)	time (s)	gap	time (s)	gap	time (s)	gap	time (s)	gap
adnoun	1.74	1.99	0	8.25	0	0.02	0	0.19	0
football	86.6	12.9	0	12.5	0	0.32	0	0.68	0
jazz	22.4	9.8	0	829.9	0	1.05	0	5.21	0
celegansm	46.1	155.3	0	>3600	0.16	1.06	0	26.9	0
email	>3600	>3600	0.21	>3600	0.17	>3600	0.07	>3600	0.07
polblogs	>3600	>3600	0.01	>3600	0.01	>3600	0.01	>3600	0
add20	>3600	>3600	0.02	>3600	0.02	160.8	0	106.4	0
data	>3600	>3600	0.32	>3600	0.22	72.5	0	84.1	0
3elt	>3600	>3600	0.46	>3600	0.33	124.1	0	133.6	0
add32	>3600	487.5	0	527.6	0	48.5	0	62.5	0
hep-th	>3600	>3600	0.12	>3600	0.12	925.7	0	1336	0
whitaker3	>3600	>3600	0.46	>3600	0.41	1372	0	1413	0

reported since for the unsolved instances the method could not find any feasible solution, nor give upper bounds for the solution in the given time limit.

For $k = 2$, $\text{SatMC}\{1, 2\}$ were the only methods which solved optimally all instances within the time limit. In all but three cases (add20, polblogs, and hep-th), their running times were significantly better than those of the other methods. Here, SatMC1 was for medium-size dense instances faster than SatMC2 , whereas the latter was better for large sparse graphs. B&B and VNS performed similarly solving efficiently small and medium-size instances.

For $k \in \{3, 4\}$, SatMC1 , followed by SatMC2 , was the best method regarding the running times, the number of solved instances, and the optimality gaps. Only

Table 4. Computational results of solving $MkCP$ for $k = 4$ on DIMACS instances.

Instance	IPMC	B&B		VNS		SatMC1		SatMC2	
	time (s)	time (s)	gap	time (s)	gap	time (s)	gap	time (s)	gap
adnoun	2.18	0.86	0	0.94	0	0.34	0	0.49	0
football	177.4	0.01	0	0.01	0	0.22	0	0.92	0
jazz	492.4	8.87	0	9.11	0	137.9	0	21.8	0
celegansm	>3600	186.1	0	194.5	0	45.3	0	95.1	0
email	>3600	>3600	0.03	>3600	1	>3600	0.61	>3600	0.51
polblogs	>3600	>3600	1	>3600	1	>3600	0.37	>3600	0.15
add20	>3600	>3600	1	>3600	1	514.8	0	594.1	0
data	>3600	>3600	0.31	>3600	0.31	112.5	0	151.4	0
3elt	>3600	>3600	0.54	>3600	1	171.1	0	186.7	0
add32	>3600	3322	0	>3600	1	66.1	0	71.2	0
hep-th	>3600	>3600	1	>3600	1	>3600	0	>3600	0
whitaker3	>3600	>3600	0.5	>3600	1	1563	0	1687	0

Table 5. Computational results of solving $MkCP$ for $k = 2$ on random instances. The number of unsolved instances is indicated in brackets.

D	n	NDV	$\bar{\omega}_2$	IPMC		VNS		SatMC1		SatMC2	
				time (s)	gap	time (s)	gap	time (s)	gap	time (s)	gap
0.10	100	min	21	18.4	0	1.66	0	0.06	0	0.08	0
		max	24.5	32.5	0	2.51	0	0.07	0	0.09	0
	150	min	26.6	3436(9)	0.26	41.1	0	1.64	0	1.84	0
		max	32.8	3600(10)	0.21	61.6	0	3.77	0	3.84	0
	200	min	33.6	3600(10)	1.37	1018	0	30.1	0	31.5	0
		max	43.1	3600(10)	0.74	2051(3)	0.05	204.5	0	176.6	0
0.15	100	min	32.3	922.1	0	59.4	0	1.03	0	1.13	0
		max	42.5	11.6	0	15.6	0	0.25	0	0.29	0
	150	min	53.3	3600(10)	0.53	3600(10)	0.26	539.8	0	575.2	0
		max	80.8	545.9(1)	0.4	429	0	23.3	0	24.2	0
	200	min	79.3	3600(10)	1.88	3600(10)	0.43	3600(10)	0.05	3600(10)	0.04
		max	124.4	107.9	0	2005(3)	0.01	1530(3)	0.01	1591(3)	0.01
0.20	100	min	65	7.74	0	61.3	0	0.99	0	1.09	0
		max	68.9	0.51	0	11.3	0	0.08	0	0.16	0
	150	min	129.8	1.48	0	111	0	0.73	0	1.11	0
		max	122.7	1.35	0	50.2	0	0.67	0	0.92	0
	200	min	192.4	3.34	0	113	0	0.13	0	0.52	0
		max	176.8	3.66	0	111	0	0.51	0	1.53	0

one large (hep-th for $k = 4$) and two medium-size instances email and polblogs could not be solved optimally by our methods; nevertheless good approximate solutions could be found. When comparing the two branch-and-bound techniques, B&B solved the same number of instances as VNS, but was faster. However, the latter delivered for $k = 3$ better approximations. Interestingly, for $\text{SatMC}\{1, 2\}$, the running times required for M3CP on medium-size graphs add20, data, 3elt, and add32 were longer but still comparable with those needed for M2CP on those instances. The worst performance across all k tested here showed IPMC.

Results for random graphs. Tables 5 and 6 present the results of solving $MkCP$ on random graphs. We restrict the results to $k \in \{2, 3\}$. Since B&B and VNS performed on random graphs tested similarly, we provide only the results of VNS, the one of a better overall performance. Average running times and average optimality gaps, both computed across the 10 graph samples of a given D , n , and NDV, are reported. Additionally, for each instance category, i.e., for a given D , n , and NDV, we provide average 2- and 3-club numbers $\bar{\omega}_2$ and $\bar{\omega}_3$, computed for each category from the 10 (optimum) values of ω_2 and ω_3 found by our methods. Noteworthy, to compute ω_2 and ω_3 , only for 13 of a total of 360 graph samples more than one hour was needed. Finally, the average optimality gap for IPMC was calculated from the gap values returned by the integer routine of CPLEX.

Table 6. Computational results of solving MkCP for $k = 3$ on random instances. The number of unsolved instances is indicated in brackets.

D	n	NDV	\bar{w}_3	IPMC		VNS		SatMC1		SatMC2	
				time (s)	gap	time (s)	gap	time (s)	gap	time (s)	gap
0.035	100	min	24	1.91	0	1.13	0	0.02	0	0.05	0
		max	27.1	1.94	0	1.31	0	0.01	0	0.06	0
	150	min	31.6	95.3	0	7.49	0	0.18	0	0.41	0
		max	33.7	213.6	0	10.4	0	0.23	0	0.49	0
	200	min	36.9	2381(4)	0.12	54.2	0	1.54	0	2.91	0
		max	40.8	3105(7)	0.18	107.8	0	2.86	0	5.08	0
0.05	100	min	30.4	7.69	0	3.25	0	0.04	0	0.11	0
		max	33.9	6.14	0	3.18	0	0.04	0	0.14	0
	150	min	43.9	2750(6)	0.06	130.2	0	1.71	0	2.96	0
		max	56	1138(2)	0.02	128.1	0	2.05	0	4.04	0
	200	min	55	3600(10)	0.77	3578(9)	0.19	91.4	0	128.7	0
		max	84.3	3168(8)	0.21	2311(3)	0.06	65.9	0	117.1	0
0.10	100	min	82.8	1.19	0	8.77	0	0.03	0	0.31	0
		max	81.8	1.36	0	8.86	0	0.03	0	0.31	0
	150	min	146	10.1	0	42.8	0	0.12	0	1.51	0
		max	141.5	13.2	0	46.8	0	0.12	0	1.46	0
	200	min	199.2	11.9	0	105.4	0	0.33	0	5.88	0
		max	196.5	13.4	0	197.2	0	0.51	0	6.32	0

For $k = 2$ and average densities $D = 0.10$ and 0.20 , **SatMC**{1, 2} found optimal solutions for every instance, whereas the running times were by far shorter than those of the other methods. For graphs of $D = 0.15$, being reportedly the hardest ones [17], our methods obtained optimal solutions except for 13 test samples of size $n = 200$, for which, however, competitive approximate solutions could be given. When comparing the performance for densities > 0.10 , **SatMC**{1, 2} solved instances with maximum NDV always faster than those of the same size and density but with minimum NDV. This could not be observed for the other methods. However, for $D = 0.15$ and all methods, the instances with minimum NDV turned out to be much harder than those with maximum NDV. IPMC was the slowest method regarding graphs of $D = 0.10$ and 0.15 , but it performed exceptionally well for $D = 0.20$, taking the third place by beating VNS.

For $k = 3$, our methods were able to solve all instances into optimality, whereas the average running times required were considerably shorter than those of the other methods. This held also for instances of challenging densities 0.035 and 0.05 according to [1, 8]. VNS exhibited the third best performance, solving optimally all but 12 instances. However, for $D = 0.10$, it was beaten by IPMC. For both values of k , **SatMC1** performed slightly better than **SatMC2**, primarily due to smaller size of the CNF encodings.

Finally, for a given k, n , and NDV, as the density D increased, the average solution size found by all methods increased, too, while the average running time

Table 7. Challenging densities D for solving $MkCP$ on random instances.

Method	$k = 2$			$k = 3$		
	$n = 100$	$n = 150$	$n = 200$	$n = 100$	$n = 150$	$n = 200$
IPMC	0.15	0.1, 0.15	0.1, 0.15	0.05	0.05	0.035, 0.05
VNS	0.15, 0.2	0.15	0.1, 0.15	0.1	0.05	0.05
SatMC{1, 2}	0.15	0.15	0.15	0.05	0.05	0.05

increased first up to a peak and then declined for higher densities. This can be explained by the fact that sparse graphs are easier to solve, because there are less possibilities to construct a k -club, whereas for higher densities many of the problems are becoming trivial. More specifically, for $D \geq 0.20$ and $k = 2$, and for $D \geq 0.10$ and $k = 3$, the values of $\overline{\omega}_k$ approached n and the instances became easier to solve despite their growing sizes. The peak average running time can be used to determine the challenging densities for an algorithm for $MkCP$ [17]. Table 7 gives those densities identified empirically for all methods tested. The numerical results show that for a given k , the challenging densities were the same for minimum and maximum NDV instances, and decreased as n increased. This effect was most evident for IPMC, followed by B&B and VNS. SatMC{1, 2} turned out to be least affected, indicating clearly their better robustness.

4 Conclusion

In this paper, we presented two PARTIAL MAX-SAT formulations of $MkCP$ for a positive integer k . Using those encodings, we implemented two exact methods for $MkCP$, SatMC1 and SatMC2, and evaluated them experimentally for typical values of $k \in \{2, 3, 4\}$ both on real-life as well as on random graphs. The computational study showed that our approach outperforms other state-of-the-art algorithms developed in the last years. It computed optimal solutions in most cases much faster and found good approximate solutions in case the computation had to be terminated. Its short running times on small and moderate-size instances qualify it clearly for usage in interactive tools, e.g., for clustering biological networks [5], providing useful insights into substructures in those networks.

It would be of interest to adapt our ideas for solving other clique relaxations like k -clique, k -plex, or R -robust k -club [4,22,24]. Moreover, for $k = 2$, our approach could also be compared with the parameterized algorithm of Hartung et al. [14], and for a general k with the branch-and-bound method by Chang et al. [11]. One could also evaluate our methods for solving $MkCP$ for $k > 4$ on power-law graphs from bioinformatics and social web applications. Finally, it is an open question, if any of the algorithmic ideas of modern CDCL SAT-solving, from which our approach clearly benefits, could successfully be extended to BIP.

Acknowledgments. The author would like to thank Shahram Shahinpour and Sergiy Butenko for providing an implementation of their method from [23].

References

1. Almeida, M.T., Carvalho, F.D.: Integer models and upper bounds for the 3-club problem. *Networks* 60(3), 155–166 (2012)
2. Asahiro, Y., Miyano, E., Samizo, K.: Approximating maximum diameter-bounded subgraphs. In: *LATIN 2010: Theoretical Informatics*. pp. 615–626. LNCS (2010)
3. Bader, D.A., Meyerhenke, H., Sanders, P., Wagner, D. (eds.): *Graph Partitioning and Graph Clustering - 10th DIMACS Implementation Challenge Workshop, 2012*, Contemporary Mathematics, vol. 588. American Mathematical Society (2013)
4. Balasundaram, B., Butenko, S., Hicks, I.V.: Clique relaxations in social network analysis: The maximum k -plex problem. *Oper Res* 59(1), 133–142 (2011)
5. Balasundaram, B., Butenko, S., Trukhanov, S.: Novel approaches for analyzing biological networks. *J Comb Opt* 10, 23–39 (2005)
6. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): *Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications*, vol. 185. IOS Press (2009)
7. Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M.: The maximum clique problem. In: *Handbook of Combinatorial Optimization*. pp. 1–74. Kluwer (1999)
8. Bourjolly, J.M., Laporte, G., Pesant, G.: An exact algorithm for the maximum k -club problem in an undirected graph. *Eur J Oper Res* 138(1), 21–28 (2002)
9. Carvalho, F.D., Almeida, M.T.: Upper bounds and heuristics for the 2-club problem. *Eur J Oper Res* 210(3), 489–494 (2011)
10. Cha, B., Iwama, K., Kambayashi, Y., Miyazaki, S.: Local search algorithms for partial MAXSAT. In: *Proc. of AAAI/IAAI*. pp. 263–268 (1997)
11. Chang, M.S., Hung, L.J., Lin, C.R., Su, P.C.: Finding large k -clubs in undirected graphs. *Computing* 95(9), 739–758 (2013)
12. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-driven answer set solving. In: *Proc. of IJCAI-07*. pp. 386–392 (2007)
13. Gendreau, M., Soriano, P., Salvail, L.: Solving the maximum clique problem using a tabu search approach. *Ann Oper Res* 41(4), 385–403 (1993)
14. Hartung, S., Komusiewicz, Ch., Nichterlein, A.: Parameterized algorithmics and computational experiments for finding 2-clubs. In: *IPEC’12*. pp. 231–241 (2012)
15. Hartung, S., Komusiewicz, Ch., Nichterlein, A.: On structural parameterizations for the 2-club problem. In: *Proc. of SOFSEM’13*. pp. 233–243 (2013)
16. IBM ILOG CPLEX Optimization Studio (2009), <http://www.ibm.com/software/integration/optimization/cplex-optimization-studio/>
17. Mahdavi Pajouh, F., Balasundaram, B.: On inclusionwise maximal and maximum cardinality k -clubs in graphs. *Discrete Optim* 9(2), 84–97 (2012)
18. Mokken, R.J.: Cliques, clubs and clans. *Quality & Quantity* 13(2), 161–173 (1979)
19. Schäfer, A.: Exact algorithms for s -club finding and related problems. Master’s thesis, Friedrich-Schiller-University, Jena (2009)
20. Schäfer, A., Komusiewicz, Ch., Moser, H., Niedermeier, R.: Parameterized computational complexity of finding small-diameter subgraphs. *Optim Lett* 6(5), 883–891 (2012)
21. Scott, J.: *Social Network Analysis: A Handbook*. Sage Publications, London (2000)
22. Seidman, S.B., Foster, B.L.: A graph-theoretic generalization of the clique concept. *J Math Soc* 6(1), 139–154 (1978)
23. Shahinpour, S., Butenko, S.: Algorithms for the maximum k -club problem in graphs. *J Comb Opt* 26(3), 520–554 (2013)
24. Veremyev, A., Boginski, V.: Identifying large robust network clusters via new compact formulations of maximum k -club problems. *Eur J Oper Res* 218(2), 316–326 (2012)